Final report

Peng Liu, Mi Zhang

4/25/2022

Abstract

The discussion of how brain neurons can affect behaviors has always been a popular topic for the study of neuropsychiatric and neurodegenerative disorders. In this project, we explore three models including the Generalized Logistic model, Principal Components Analysis, and Recurrent Neural Network using the data of Elevated Zero Maze for mice behavior prediction based on their neuron frequencies. In addition, we aim to compare the advantages and disadvantages of using different models and discuss possible methods to improve model accuracy and the avoid the problem of overfitting.

Introduction

Social deficits are the first signs of many neuropsychiatric and neurodegenerative disorders, including schizophrenia, major depressive disorder, Alzheimer's disease, and frontotemporal dementia, which lead to progressive social dysfunction. Our client, who wanted to understand how brain circuits control behavior, tried to record data on neural circuits as mice socialized. The experiments were divided into three groups, Direct Interaction, Opposite Sex, and elevated zero maze. We choose the elevated zero maze set of data as our analysis object. In this set of experiments, mice were placed in a closed loop for ten minutes, and the maze was divided into two areas, open arm and close arm. Open made the mice anxious, and close was the opposite. The researchers recorded neuronal data from mice in different arms. There were a total of 13 mice in this set of experiments, and we selected the first six for our study. All data are divided into two types, behavior, where column 1 indicates when the animal is in the closed arm, and column 2 indicates when the animal is in the open arm. The other is z_score, which records single-cell records every 0.1 seconds. In this project, we aim to explore and compare different models for the prediction of mice behaviors using the frequencies of different neurons.

EDA

First of all, we transposed binned behaviors data and combined them with all neuron z-scores as one data frame. And then, we want to understand the relationship between binned behaviors and frequency changes of different neurons in different mice, so we randomly pick the first neuron of each mouse to plot line-point plots. As we can see in figure 1, all six mice show no interaction at the beginning of the experiment. It is easy to see in mouse_409, neuron_1 shows high frequencies in the open arm. However, for mouse_412, mouse_417, and mouse_418, they seem to stay in the closed arm longer than in the open arm and do not show much of the frequency changes throughout the experiment. On the other hand, there are even changes in closed arm and open arm for mouse_414 and mouse_416. Since our client had mentioned that all neurons are randomly picked and assigned, we can not conclude similarities and differences in those neurons.

Figure 1: EDA of Six Mice

Model Fitting

Before we start our model fitting, we decide to drop the rows containing zeros in both columns of binned behaviors since those rows show that mice do not have any interaction. The data splitting ratio is 70% for training and 30% for testing. Furthermore, we use the confusion matrix to evaluate the model accuracy for all three of the models.

GLM

We first tried to build a baseline model. We start with a basic Generalized Logistic model, which is a logistic model. We choose the closed arm as our output, and all neuron z-scores as input. By looking at the Coefficients of glm, you can find that there are many variables in the input that are meaningless.

We then predicted the test dataset and evaluated the predicted results as shown in Table 1. From the table, we can see that except for the mouse_414 and mouse_416, the accuracy and AUC_score of other mice are almost all above 90%, especially mouse_412 and mouse_417.

Table 1: Comparison of GLMs for Six Mice

Mice	Total_neurons	Accuracy	AUC_score
409	110	0.8955	0.9582
412	100	0.9403	0.9623
414	33	0.7373	0.7870
416	26	0.7271	0.7852
417	79	0.9476	0.9684
418	77	0.9168	0.9505

GLM after PCA

After fitting the baseline model, we try to perform principal component analysis on the behavior data, reduce the dimensionality of the data, reduce the number of variables to a large extent, and then bring the data back to the baseline model for analysis.

Before conducting principal component analysis, we first performed a KMO test on the data (Table 3) to ensure that the data were suitable for principal component analysis. From the table, we can find that although the KMOs of six mice are all higher than 0.5, the mice_416 is the only one lower than 0.7. Of the other five mice, mice_409 had strong correlations, and the other four had strong or moderate correlations. Therefore, these six data all support PCA. Next, we use the Parallel test and scree plot as shown in Figure 2 to determine the number of principal components. Through the gravel plot, we set the number of principal components as the number of scatter points above the red dotted line. The red dotted line represents the average eigenvalue of the random data matrix. When the number of principal components or factors increases, the eigenvalues of the real data are lower than the average eigenvalues of random data, which means that the following factors or principal components have no retained value.

```
## Parallel analysis suggests that the number of factors =  NA  and the number of components =  20
```

Finally, we extract the required principal components as shown in Table 2 and re-merge it with the binned behaviors columns and bring it back into the baseline model. We then generate confusion matrix and plot ROC curve with given AUC_Score to evaluate model accuracy as in GLM (Table 3). The results are also very similar to GLM, except for the mouse_414 and mouse_416, the accuracy of other mice is higher than 80, and the only difference is that in GLM after PCA, only the mouse_417 has an accuracy higher than 90.

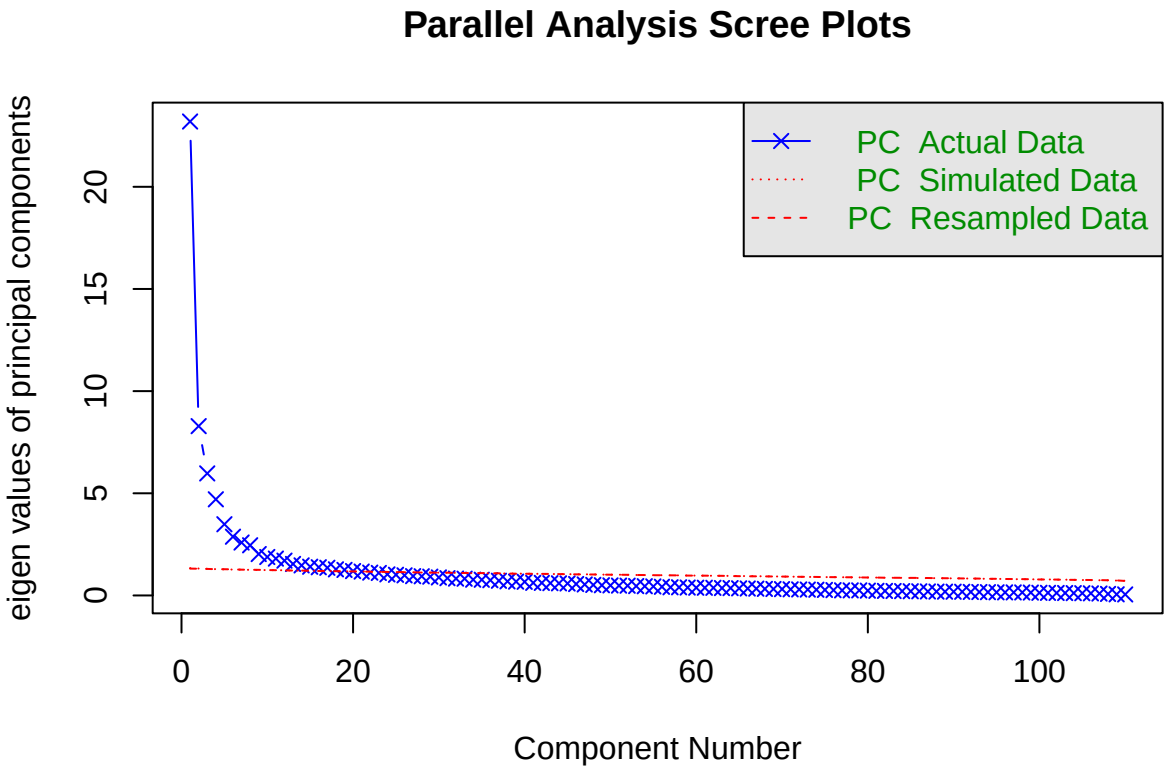

Figure 2: Parallel Analysis Scree Plot

Table 2: First 5 row of principal components

	PC1	PC2	PC3	PC4	PC5
binned.behavior.74	0.3975876	-1.1137939	2.1372607	-3.274198	0.9412561
binned.behavior.98	4.3816029	0.5942148	-0.5109036	-2.762968	-1.1241985
binned.behavior.99	4.7326636	0.9258599	-0.6580597	-2.835772	-1.0716136
binned.behavior.100	1.7071796	0.3736926	-0.1587083	-3.370195	-0.7906298
binned.behavior.101	-1.9364478	-1.2035017	0.7539303	-4.024487	-0.1753720

Table 3: Comparison of GLM after PCA

Mice	KMO	PC_number	Accuracy	AUC_score
409	93	24	0.8109	0.8710
412	77	27	0.8385	0.8520
414	80	11	0.6587	0.7069
416	61	11	0.7033	0.7365
417	73	24	0.9012	0.9015
418	78	24	0.8573	0.8737

RNN

Besides Generalized Logistic model and Principal Component analysis, we also want to explore Recurrent Neural Networks for our sequential data. Noticing that we have 6300 rows of time sequence which is 10 minutes long, We decide to explore Long Term and Short Term Memory (LSTM) in which it "overcomes the problem of early signals being washed out by the time they get propagated through the chain to the final activation vector" [1].

To set up our model, we use the frequency of neurons for prediction, and predicting the outcome of binned behaviors. Our model contains five layers including two gated recurrent unit layers, two dropout layers with 0.2 dropout rate to avoid overfitting, and one fully connected layer. The first layer has 128 units along with the dropout layer, and the second layer has 64 units with the dropout layer. Both recurrent layers use "ReLu" activation. The last layer has two units and uses"sigmoid" activation which converts linear function into probability between zero and one. For regularization, we choose "binary_crossentropy" as the loss function since our predicting outcome is binary and "rmsprop" as the optimizer.

The most complicated part of this RNN model is to choose different combinations of hyper-parameters. Table 4 shows the number of batch_size and epochs used for six mice along with accuracy and loss. We can see that the RNN model demonstrates higher accuracy than the other two models, but we also encounter the problem of overfitting.

Table 4: Comparison of RNN for Six Mice

Mice	neuron_number	Batch_size	Epoch	Accuracy	loss
409	110	20	15	0.9533	0.2100
412	100	15	10	0.9886	0.0458
414	33	10	15	0.8937	0.4082
416	26	15	15	0.9270	0.2276
417	79	15	10	0.9728	0.1358
418	77	15	15	0.9597	0.1632

Result and Discussion

In order to compare the accuracy of prediction generated by three different models, we created a comparison plot as shown in Figure 3. It is obvious that the RNN model demonstrates higher accuracy for all six mice than other two models. In addition, both GLM and RNN models use all neuron frequencies for prediction, but RNN has higher accuracy for predicting such sequential data. Among three models, mouse_414 and mouse_416 show lower accuracy which can be explained by the lower number of total neurons than other four mice.

Despite the advantages of high accuracy using the RNN model, the model itself is constrained by the calculational expense of time and the problem of overfitting. To avoid overfitting, we can decrease the

number of units in the gated layers and increase the dropout rate. However, the decrease in units of gated layer can lead to the trade-off of the decrease in accuracy.

Figure 3: Comparison of Three Models

Conclusion

Throughout the project, we explore the most commonly used interpretable model like Generalized Logistic Model, machine learning model like Principal Components Analysis, and the novel deep learning model like Recurrent Neural Network. Those models have the same target as predicting the mice's behavior in the Elevated Zero Maze according to frequencies of different neurons. On the other hand, we realize the number of neurons used in the models also plays a big role to increase prediction accuracy. As a matter of fact, none of those models are perfect and we have to learn how to use the advantages of each model with trade-offs. Due to lack of knowledge in time series and forecasting, we hope we can apply the AutoRegression and Integrated Moving Average model to the data in the future.

Reference

[1] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning : with Applications in R. New York :Springer, 2013. APA.

[2] https://cloud.tencent.com/developer/article/1921634

[3] https://www.r-bloggers.com/2018/11/lstm-with-keras-tensorflow/